Image-Text Alignment: CLIP Training and Applications

Ammon Brock Matthew Gabbitas Yu-Hsien Jen

Figure 1. **Application**. Demonstration of the semantic lines application for image/text embeddings. Two semantically opposite captions are embedded and then one subtracted from the other to define a semantic line in embedding space. The line can then be used to rank images according to the caption heuristic. This demonstration with age ranking achieved a Spearman correlation of 0.703 with a P-value less than 0.01.

Abstract

In this project, we explore applications for vision + text aligned models. We first demonstrate the training of a CLIP-inspired model [1] by aligning two pre-trained encoders. After training with 1.2 million images from the Conceptual Captions dataset [2] for 32 hours on one P100 GPU, we were able to surpass the original CLIP model on several benchmarks and approach it on one other. We note that this performance came after fine-tuning on a much smaller dataset and with a much lower parameter-count than the original CLIP. Though part of this performance is due to the selection of data for the benchmark, we also attribute it to the robustness of the pre-trained features that each encoder started out with. We demonstrate the utilization of the aligned embedding space for five applications: Themed photo album generation, image ordering with semantic lines, embedding-space arithmetic, social media hashtag recommendation, and open-vocabulary localization. For each task, we lay out the necessary improvements and modifications to the model as well as strengths and weaknesses of the implementations explored here.

1. Introduction

Contrastive Language–Image Pre-training (CLIP) is a model developed by OpenAI that improves the latent space by aligning image embeddings with text embeddings. Specifically, CLIP learns a shared embedding space for both images and textual descriptions. The text encoder is often a Transformer-based model such as BERT, while the image encoder is often a ResNet or ViT, both of which were tried used in the original paper. By projecting images and texts into this joint embedding space, CLIP enables zero-shot classification and image retrieval without requiring task-specific fine-tuning.

2. Related Works

The model training and applications presented in this paper build upon the foundational work by [1] in demonstrating the power of contrastive learning for aligning image and text representations. Our specific training method was investigated by [3] though on a different dataset. Many of our ideas for applications were inspired by foundational works in text embeddings including [4], [5], [6], and [7].

Since CLIP is trained using a contrastive learning objective to produce a meaningful latent space, we first investigated whether vector arithmetic—well-known in word embeddings such as GloVe (e.g., $king – man + woman \approx queen$)—would exhibit similar behavior in CLIP embeddings. However, our preliminary experiments showed that such vector math does not translate well to CLIP's visual embeddings. When we averaged or subtracted two image embeddings and attempted to retrieve the nearest image, the results were essentially unrelated to the original inputs. This motivated us to explore more principled approaches for improving or restructuring latent spaces.

One relevant line of research is the whitening transformation proposed by Zhi Chen, Yijie Bei, and Cynthia Rudin. Their method was originally applied to convolutional neural networks (CNNs) to improve model interpretability.[8] Their key idea is that the latent representations learned by CNNs are often non-centered and correlated across dimensions. Whitening addresses this by standardizing features to have zero mean, unit variance, and no cross-dimension correlation, making the latent space more

structured.

Building on this concept, Roy Betser, Meir Yossef Levi, and Guy Gilboa further extended whitening to CLIP embeddings. [7] Their work introduced Whitened CLIP, which applies an invertible linear transformation to CLIP's latent space. After whitening, the embeddings lie on a more uniform hyperspherical distribution, enabling improved estimation of image likelihood. Their results demonstrated that this whitened space can be used for detecting quality issues in AI-generated images. Importantly, they also showed that angular distances between whitened embeddings more reliably reflect semantic differences between images.

Inspired by these advancements, we incorporate whitening-based latent space refinement into our system as well. The details of our implementation will be discussed in 4.3.

We implement open-vocabulary localization, the details of which are described in 4.5. This implementation was inspired by the work by [6], which used CLIP as a backbone for performing segmentation given text or image prompts.

3. Training And Evaluation

3.1. Training Process

We followed the training process outlined by [3] to align two pre-trained encoders, an image coder and a text encoder respectively. For the image encoder, we used ViT with 16x16 patches and 224x224 image resolution. For the text encoder, we used standard Bert. Our complete model had 196 million parameters.

We made our train set of 1.2 million images from Google's Conceptual Captions dataset [2]. Using a batch-size of 32, learning rate of 5e-6, and cosine LR scheduler with warm-up, we trained for 4 epochs (32 hours) on a single P100 GPU, driving contrastive loss down near 0.2.

Implementation was fairly straightforward with the VisionTextDualEncoder class from Hugging Face that handled the addition of projection heads onto our pretrained encoders. The majority of the work was loading and formatting the data for efficient training on such a large dataset.

3.2. Evaluation and Benchmarks

We evaluated our model and compared it to CLIP with the 4 metrics displayed in Table 1. The purpose of the evaluation was to verify that the training was done correctly and not necessarily prove superiority over CLIP. However, our model somewhat surprisingly outperformed CLIP on several tasks. A direction for future work could be more wholistic evaluations and comparisons with CLIP and state-of-the-art methods. The metrics used in Table 1 are defined below.

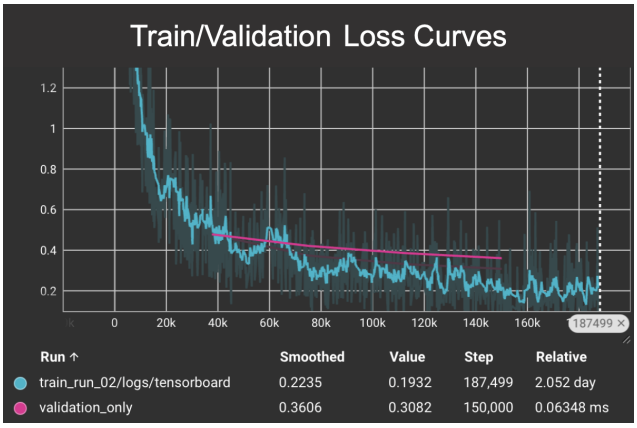

Figure 2. Training and validation loss curves.

	Our Model	CLIP
Recall@10	**0.1650**	0.0045
Median Rank	**129**	23700
CIFAR-10 Zero Shot Accuracy	0.9114	**0.9571**
Semantic Age Ranking Correlation	**0.7027**	0.6479

Table 1. Our trained model outperforms CLIP on 3 of our 4 evaluation metrics.

Recall@10 is defined as:

$$\text{Recall@10} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 & \text{if rank}(y_i \mid x_i) \leq 10 \\ 0 & \text{otherwise}\end{cases} \quad (1)$$

where N is the number of caption-image pairs (x_i, y_i) in a dataset and $\text{rank}(y_j \mid x_k)$ is computed by calculating the distance between x_k and all N images $\{y_1, \ldots, y_N\}$ in the dataset, then determining the rank of y_j among these distances. In English, this measures how often a ground truth image is one of the 10 closest images to a given caption. We note that this metric heavily depends on the dataset size. It is much easier to obtain a higher recall@10 when $N = 50$ than when $N = 116,000$. For our comparison, we used a hold out dataset of size $116,000$, hence the low recall@10 scores. Our model shows a much better recall@10 score than CLIP. We believe that a likely explanation has to do with the distribution of training images and captions. The captions in our training may more closely align with the types of captions in the hold out set than the captions in the original CLIP training data.

Median Rank is another measure of the model's ability to query a dataset of images given a caption as a query. $\text{rank}(y_i \mid x_i)$ is calculated for each caption-image pair, and the median of all pairs in the dataset is recorded. This metric was recorded on the same dataset as recall@10 with $N = 116,000$. As with recall@10, the median rank metric grows in difficulty as the size of the dataset grows.

Our model significantly outperforms CLIP on this metric as well.

CIFAR-10 Zero Shot Accuracy. Following the example of [1], we were able to measure the classification accuracy of our model and CLIP despite never having trained these models for classification. For a given model, we use its image encoder to compute image embeddings for each image in the CIFAR-10 dataset [9]. Then we create a text embedding for the sentence 'A photo of {label}' for each of the 10 classification labels. To classify an image, we simply compute the cosine similarity between the image embedding and each text-label embedding and classify the image according to the label with the highest similarity. This allows us to compute zero shot classification accuracy. This was the only metric where CLIP outperformed our model. We suspect that captions of the format 'A photo of {label}' more closely align with the distribution of captions that the original CLIP model was trained on. It may be that CLIP outperformed our model on this task simply because the method of converting a class label to a caption was originally designed to improve the performance of CLIP. In the future, it would be interesting to investigate different formats for converting labels into captions and observe these effects on zero shot classification accuracy.

Semantic Age Ranking Correlation measures a model's ability to sort images based on semantic directions defined by pairs of captions. Because performance on a task like this is hard to quantify in a general sense, we narrowed our focus to a specific semantically defined direction with a ground truth we could use for comparison – age. We used each model's embeddings to compute age rankings for a sample of 1000 images on the UTKFace dataset [10]. Then we computed the Spearman correlation between a model's ranking and the true age ranking given by the ages in the dataset. Our model outperformed CLIP on this metric as well. Importantly, the data used for the computation of this metric was entirely out of the distribution of our model's training data, yet our model still performed strongly. More details on the implementation of this metric can be found in 4.2.

4. Application Experiments

We implemented several task-specific applications using our CLIP model as a backbone.

4.1. Album Generation

For data preparation, we organized 600 images into a folder, primarily sourced from a personal Google album containing photos of the author and family visiting Japan and various attractions. Users can substitute their own images if desired. Each image was processed using a trained CLIP model to generate embeddings, which were then stored in a FAISS vector database for similarity search. The backend server, implemented in Python using Flask, provides an endpoint that takes a text query, generates its corresponding text embedding, searches the FAISS database for the 16 most similar images, and returns the filenames of these images to the client. On the frontend, we built a React-based interface that displays thumbnail images to optimize loading speed and provides an interactive user experience. Additionally, the frontend uses JavaScript to dynamically generate image albums when the user clicks a "Generate Album" button; the album automatically organizes the 16 closest images into a single PDF file and downloads it to the user's local storage.

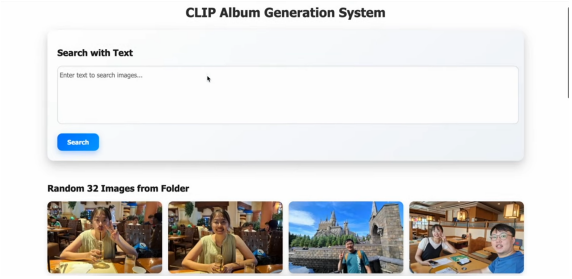

Figure 3. Screenshot of the album generation system. The bottom part shows random images selected from folders, while the top part contains a text search input area.

4.2. Image Ordering With Semantic Lines

Semantic lines can be easily defined in text embedding space by subtracting pairs of captions with opposite meanings. We were curious if image-text alignment with contrastive loss would also align these semantic directions in the shared representation space. One way to investigate this is to project a model's embeddings for a set of images onto a semantic line defined by a pair of captions and observe the quality of the ordering induced by the projection.

More formally, given two captions, a model's text encoder produces vector representations x_1 and x_2 for each caption. The difference $x_2 - x_1$ defines a vector s_l that points from x_1 to x_2, which we call the *semantic line* defined by the given captions. A model's encoding of an image y_i can then be projected onto s_l:

$$\text{proj}_{s_l}(y_i) = c \cdot s_l \tag{2}$$

where $c \in \mathbb{R}$ is a scalar coefficient. We can order all images in a dataset by their corresponding c values after projection onto s_l. Ideally, images with the highest c values would be "more like" the caption represented by x_2, while images with the lowest c values would align more closely with x_1, with a smooth transition in between.

Note that this process can be done for any pair of captions and for any dataset of images. This means images can be easily sorted from "a happy person" to "a sad person", from "a dumb thing to do" to "a smart thing to do", from "a

fraudulent check" to "a valid check", or countless other semantic directions. The quality of these orderings obviously varies based on the quality of the model and how well defined the semantic direction is in the model's representation space.

For our experiments, we used a semantic line with an associated ground truth so that we could quantitatively evaluate our model's success at sorting images according to the semantic line. The captions "An extremely old man" and "A very young baby" define define the semantic line of age. The UTKFace dataset has thousands of images of faces along with the true ages in years of each person [10]. We projected a random sample of 1000 images from this dataset onto the age line to obtain a predicted age ranking. We then measured the similarity between the predicted rankings and the true rankings with Spearman correlation. A correlation of 1 would indicate perfect ranking performance while 0 would indicate no relation between predicted rankings and true rankings. We evaluated the semantic age ranking correlation for both our model and CLIP and the results are listed in Table 1.

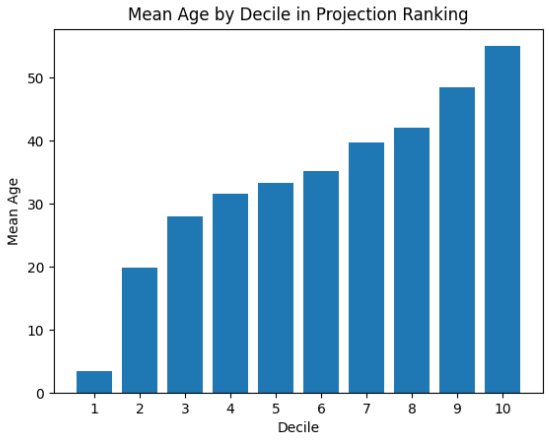

Figure 4. The mean age of each decile of our model's predicted age ranking. The mean age increases with each decile, indicating strong ranking performance.

To visualize our model's performance on this task, after ordering the images with our model from baby to old person, we split the result into 10 groups. We then randomly sampled one image from each group and display the results in Figure 1. We also measured the mean age of each group and displayed the results in Figure 4. We note that the mean age increases for each decile of our model's ranking. This shows strong performance of our model at ranking images of faces without any age or face specific training. High scores on zero shot classification accuracy would lead us to believe that these models would be able to identify babies and elderly people, but these results show not only correct identification of those groups but also accurate sorting of the images into a smooth transition between groups.

During our experiments of ordering images with semantic lines, we were reminded of the serious implications of any racial, ethnic, or gender biases that might be associated with our model. Does race influence where a picture lands on age ranking? What happens when you sort images from smart to dumb? We consider these questions extremely important and plan on investigating this topic in the future.

4.3. Vector Math

The original idea for this application was that CLIP embeddings might support vector arithmetic in a manner similar to word embeddings such as GloVe (e.g., $king – man + woman \approx queen$), as discussed in the previous section on related work. However, experiments showed that the raw CLIP embeddings performed poorly on such vector-arithmetic tasks. To address this limitation, we applied a whitening transformation to the CLIP latent space, which decorrelates the embedding dimensions and normalizes their variances, aiming to improve the embeddings' suitability for vector arithmetic.

We implemented the whitening procedure following the work of Betser, Levi, and Gilboa. [7] Given a set of random vectors X of size $n \times d$, we first compute the mean μ and the covariance matrix Σ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \Sigma = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)(X_i - \mu)^\top.$$

Performing eigenvalue decomposition on Σ gives

$$\Sigma = V \Lambda V^\top,$$

where V contains the eigenvectors and Λ is the diagonal matrix of eigenvalues. The whitening matrix W is then defined as

$$W = \Lambda^{-\frac{1}{2}} V^\top.$$

To obtain the whitened data Y, we first center the original data $\tilde{X} = X - \mu$, and then apply the whitening transformation:

$$Y = W\tilde{X}.$$

Before performing whitening, we first applied PCA to reduce the original 768-dimensional embeddings to 256 dimensions, with the aim of reducing noise. After generating the whitened embeddings, we stored them in a vector database, associating each embedding with its corresponding original image. We then conducted two types of vector arithmetic experiments—averaging and subtraction. Given two images, we computed both the average and the difference of their embeddings and retrieved the closest image in the embedding space.

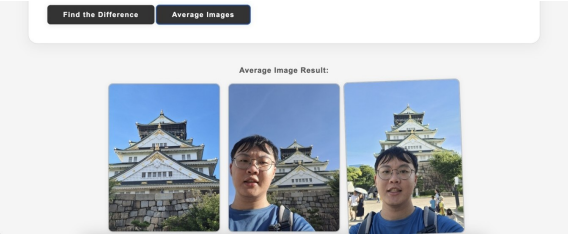

Figure 5. Vector arithmetic example. Averaging the embeddings retrieves a third image showing the same man standing in front of the castle, demonstrating that the averaged representation captures the shared semantic content.

We found that averaging works well in practice. For example, computing the average of two images—one of a person standing in front of a castle and the other of just the castle—produced results dominated by images of the castle, some including a person and some without, which we consider satisfactory (see Fig. 5). In contrast, subtraction produced unstable results. Subtracting two embeddings often pointed the vector to a random region of the latent space, yielding retrievals that were largely uninterpretable. Our hypothesis is that this occurs because CLIP is trained primarily for image–text alignment rather than for encoding sentiment or other decomposable semantic attributes. Averaging works because similar images tend to have similar textual meanings, so the averaged embedding remains in a semantically reasonable region of the space. Subtraction and addition, however, implicitly assume a well-defined linear semantic structure, which CLIP is not explicitly trained to support.

4.4. Hashtag Generation

We used the embeddings from our model to build a tool to automatically recommend hashtags for social media images. For data collection, we sourced hashtags from a PDF book available online titled "The Ultimate List of Popular Instagram Hashtags," published by the company HeyOrca. The book provides both the top 100 general hashtags and a variety of category-specific hashtags covering topics such as food, hair, health & fitness, music, travel, and lifestyle. To extract these hashtags, we used Python with regular expressions, enabling the automated parsing of the book's structured lists into a usable dataset. After extraction, each hashtag was encoded into a vector using a pretrained CLIP text encoder; these embeddings were then stored in a FAISS vector database. In total, we collected 1,460 hashtags, forming the basis for our hashtag recommendation system.

On the backend, we implemented a server using the Python Flask framework, which exposes an endpoint to receive an input image from the client. Upon receiving the image, the server computes its embedding using the pretrained CLIP image encoder and searches the FAISS database to retrieve the 10 most semantically similar hashtag embeddings. These top-10 results are returned to the client in real time.

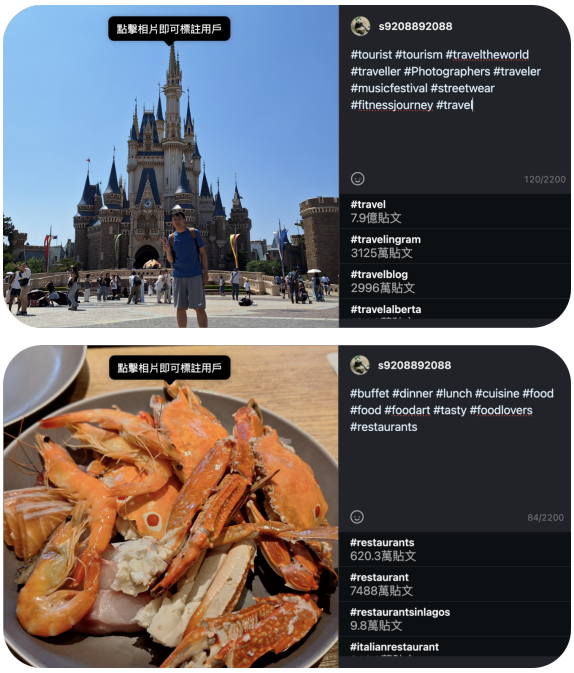

Figure 6. Examples of Instagram hashtag recommendation. The example images show that our trained CLIP model correctly identifies the Disney castle image with the travel-related text, and the food image with the corresponding food-related text.

To provide a seamless user experience, we developed a Chrome Extension for Instagram. The extension features a button that users can click after uploading a photo on Instagram. When activated, it captures the uploaded image, sends it to the backend server for processing, and subsequently displays the recommended hashtags returned in the response. This design allows users to obtain contextually relevant hashtags for each image automatically, eliminating the need for manual keyword searches or input and streamlining the hashtag recommendation process.

4.5. Open Vocabulary Localization

We also utilized the aligned embedding space to achieve zero-shot image localization. For the localization task, we compute embeddings for a natural language query term as well as for every patch of the image. We then measure patch similarity to the query term using cosine similarity. We upsample the similarity scores back to the image dimension and generate a heatmap over the image showing the location of the target class within the image. Multiple objects can be localized within the image by calculating patch-wise cosine similarity for every object description and running soft-max across the class dimension to generate complementary heatmaps for every object. A visualization of these

results is shown in figure 7.

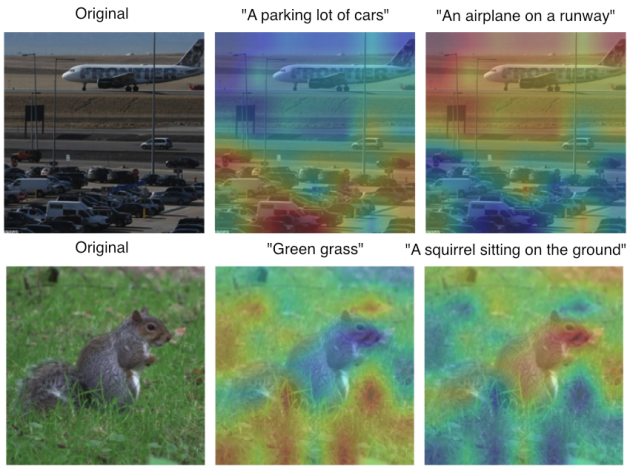

Figure 7. Examples of localization using primary objects in two different photos. Despite a bit of noise, the model shows a strong ability to differentiate key classes within the image.

We note that with the addition of a segmentation head and fine tuning on a segmentation dataset, this localization ability can easily generalize to segmentation as demonstrated by [6].

5. Future Work

This project demonstrated the versatility of fine-tuning CLIP models from pre-trained encoders, showing strong benchmark performance with lower parameter-count and fewer image-text paired data than the original CLIP. We also demonstrated the application of our models features to a diverse set of tasks that benefit from the rich shared image-text embedding space.

In the process of this project, we identified several potential future directions for expanding on this work.

First, shared image-text embedding is a valuable asset for explainable AI. Techniques like [11] follow gradients to highlight parts of an input image that most contribute to the output. Conditioning this feature attribution on text prompts for a model with a text-aligned feature-space could be a valuable tool for explainability.

We mentioned in 4.5 that the addition of a segmentation head and additional training could extend the localization abilities of our model to perform open-vocabulary segmentation as well. It would be interesting to test these abilities and see how well this segmentation approach generalizes to domains that are significantly out of distribution like medical images.

We also note that any model trained on a large dataset may have unforeseen and unwanted biases. Racial or gender biases may be particularly important to investigate for the application of sorting images according to semantic lines discussed in 4.2.

Finally, this work created a shared image-text embedding space using a contrastive loss function, as used in the original CLIP paper. However, recent methods like [12] and its successors have successfully trained unsupervised features using non-contrastive methods. Research suggests that alternate unsupervised methods could surpass contrastive ones in training efficiency. It would be enlightening to compare performance of this same model trained using a non-contrastive method.

6. Note on AI Usage

Throughout the course of this project, we used various LLMs (Claude, Gemini, ChatGPT) as coding assistants. We structured the code and designed the experiments ourselves and turned to LLMs for documentation questions, function completions, questions about best practices, etc. We do not believe our usage of AI inhibited our understanding of the experiments we ran and should not cast doubt on the results we obtained.

References

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763. 1, 3

[2] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2556–2565. 1, 2

[3] X. Zhai, X. Wang, B. Mustafa, A. Steiner, D. Keysers, A. Kolesnikov, and L. Beyer, "Lit: Zero-shot transfer with locked-image text tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 123–18 133. 1, 2

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543. 1

[5] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *Advances in neural information processing systems*, 2016, pp. 4349–4357. 1

[6] T. Lüddecke and A. Ecker, "Image segmentation using text and image prompts," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 7086–7096. 1, 2, 6

[7] R. Betser, M. Y. Levi, and G. Gilboa, "Whitened clip as a likelihood surrogate of images and captions," *arXiv preprint arXiv:2505.06934*, 2025. 1, 2, 4

[8] Z. Chen, Y. Bei, and C. Rudin, "Concept whitening for interpretable image recognition," *arXiv preprint arXiv:2002.01650*, 2020. 1

[9] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. 3

[10] Z. Zhang, Y. Song, and H. Qi, "Age and gender classification using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 34–42. 3, 4

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626. 6

[12] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660. 6